BPM Data Collection for MTA

Local application

Thu, May 3, 2007

Beam Profile Monitors for the MuCool Test Area area are interfaced via ethernet. A means of collecting this network-based data is desired for assimilation with other Linac data of a PowerPC front end. This note describes a method of reading this data into the front end's data pool.

PowerPC front ends use the VxWorks operating system. The basic approach here is to add a new task to the system that acts as a receiver for the reply data from the BPM hardware. This allows us to operate this particular form of network communications during the time that the `Update` task is running to fill the data pool with the most recent readings for the present 15 Hz cycle.

Each 15 Hz cycle, beginning at a time called "μP Start," the `Update` task is activated to update its local data pool, which it does by interpreting a list of instructions found in the Data Access Table. One of these instructions causes all active local applications to be invoked to do what they need to do for that cycle. One of these LAs, perhaps called `BPMD`, can help support this BPM data collection transaction by issuing a data request that targets a special UDP port in the BPM hardware. It waits until the reply data has been received and processed by the new higher priority task, then delivers the fresh BPM data readings into a sequence of channels in the data pool. After all LAs have run, it fulfills all active data requests before releasing the cpu to let the other same-priority tasks run.

The new higher priority task, perhaps called `BPMData`, is very simple. It merely waits on the file descriptor used by this BPM transaction, and whenever a reply is received, it copies the data into a place that the LA knows about, marking it as fresh. The task then loops back to await the next reply, which is expected on the next cycle. Since the task has a higher priority than `Update`, it has no trouble processing this data even while `Update` has not yet completed its work.

In order to make this work, it is obvious that information needs to be held in common between the new task and the the new LA. The task needs to know the file descriptor for which it should issue the `recvfrom()` call. It may even know the location and structure of the LA static memory block. Functionally, the task is an extension of the LA.

Parameter layout

Here are the parameters used by `BPMD`:

Prompt	*Size*	*Meaning*
`ENABLE B`	2	Usual LA enable Bit#
`REPLY`	2	Time to await reply, in ms
`VALID B`	2	Valid data status Bit#
`BPMDATA C`	2	BPM data base Chan#
`NBPMS`	2	#BPMs
`IPADDR`	4	IP address of BPM hardware

An enhancement to get the data request sent more promptly is to support a new variation on the "Run all LAs" instruction for the Data Access Table. A spare word could specify a (nonzero) `LATBL` index that will denote a single LA instance to invoke. This allows us to place such an instruction early in the DAT, so that the request is sent right away. Later, when all LAs are called, this instance gets a second call, which gives it a chance to await the reply, assuming that it has not already arrived. Determine whether the call is first or second by monitoring the global cycle counter.